Hsueh, Peter EXHIBIT A

From: Alexander Tormasov [tor@sw-soft.com]
Sent: Thursday, July 24, 2003 1:58 AM
To: Hsueh, Peter
Cc: ASutanandi@OMM.com; Canning, C. Kelley; Berliner, Brian
Subject: Re: Patent Application (Official)

inventors are the same as for previous patent - Tormasov, Protassov and Beloussov
You should already has all info about.

Also, I have a set of signed power of attorney documents for patents (you send me some times ago).
Have I send originals using fast mail or can send you a scanned images and send originals using plain mail?

----- Original Message -----
From: "Hsueh, Peter" <PHsueh@OMM.com>
To: "'Alexander Tormasov'" <tor@sw-soft.com>
Cc: "Sutanandi, Alice" <ASutanandi@OMM.com>; "Canning, C. Kelley" <ckcanning@OMM.com>
Sent: Thursday, July 24, 2003 1:19 AM
Subject: RE: Patent Application (Official)

\> Enclosed is a copy of the official draft application. [redacted] We will
\> continue to keep you advised as to the progress of this application. In the
\> meantime, should you have any questions please do not hesitate to call me.
\>
[redacted]
\>
\> Thanks again, Peter.
\>
\>
\>
\>
\>

Method of Implementation of Data Storage Quota

1. Method of implementation of a data storage procedure, which includes:
   - a computer system, which includes authorized users who get unique IDs within any context of the operating system;
   - a user group consisting of users mentioned above who have their own unique IDs inside any context of the operating system;
   - hierarchic computer file system organized on top of one or several data storage devices, where files are arranged into trees and where each file has an ID of a mentioned above authorized user or user group as an attribute which denotes belonging to this user or group;
   - parameters of the mentioned computer file system, describing qualitative characteristics of the level of consumption of the file system resources by mentioned users and user groups;
   - a procedure of mounting of a specific data storage area as a file system inside any available directory of the file system after execution of which the mentioned computer system gets an opportunity to use the new tree of the file system as an extension of an existing one;
   - a set of mentioned directories – mounting points of the file system inside the mentioned computer file system where both file system volumes and file system sub-trees can be used as mounting objects;
   - a system of calculation of the used space associated with the mentioned files and specific area that can tell each user a total size of files, marked by the mentioned user ID in the given area, as well as other quota parameters of occupied space.

2. System of calculation of occupied space of Claim 1, which includes the following steps:
   - definition of the calculation area of occupied space as one or several sub-trees of the file system located on the search path below any directories;
   - usage of an algorithm of analysis of the full access path to a file to define belonging of the file to the mentioned area; that is, belonging to the area is defined by the presence of the address of any of the mentioned directories starting from the directory root in the full access path;
   - association of a file with the unique user or group ID of Claim 1 via usage of information stored in the file system as information about file ownership;
   - usage of a special file or data storage area or computer operating memory or special server which stores and updates data about current size of occupied space or other quantitative parameters of storage resources consumption, associated with the unique ID of Claim 1;
   - usage of a special program of the operating system or operating system kernel to serve requests for data allocation in the storage device and answers to reference requests about sizes of allowed and current occupied area of the data storage by a consumer, associated with user or group ID of Claim 1;
   - usage of the mentioned special file or data storage area or computer operating memory or special network server by the mentioned kernel programs and operating system services to define an opportunity to reserve space for user data, identified by a unique ID of Claim 1;
3. Method of usage of the system of calculation of occupied space of Claim 2, which includes:
   - method of initialization of a special file or data storage area or computer operating

memory or special server for storage and update of information about current size of occupied space of Claim 2, that defines initial values for usage parameters of the data storage space for unique user or group IDs of Claim 1 inside the area of occupied space calculation of Claim 2;
- method of setting up values of usage limits of data storage space for each unique user or group ID of Claim 1;
- method of tracking of quantitative characteristics of alteration of the resource consummation level;
- method of usage of special programs of the operating system or operating system kernel to serve requests of Claim 2, which uses mentioned pre-defined parameters for managing procedures of allocation of data storage area of Claim 1;

4. Method of allocation or release or modification of the allocated area size and other data storage parameters of Claim 1, performed by the mentioned special programs of the operating system or its kernel and consisting of the steps:
   - defining of belonging of the mentioned space to any file of the file system of Claim 1;
   - determination of belonging of the area of occupied space calculation by using the method of Claim 2;
   - determination of the unique user or group ID of Claim 1 for the mentioned file;
   - detection of the current values of storage usage parameters and ID limitations, recorded in a special file or data storage area or computer operating memory or special server of Claim 2;
   - comparison of the mentioned limitations with the current value of the storage usage, current state of the computer system, current state of the file system and requested size for allocated space and other parameters of the data storage;
   - taking of a decision about permission to the required operation based on the mentioned comparison and execution of the required operation;

5. Context of the operating system of Claim 1 where authorized users of the computer system get unique IDs, which includes:
   - a set of IDs unique in the computer network, or
   - a set of IDs unique on the given computer,
   - a set of IDs unique in the given allocated area of a computer, including ‘chroot’ and virtual environments;

6. Consumption parameters of the file system of the computer of Claim 1, which include at least one of the following:
   - size of the data storage associated to a user or user group of Claim 1;
   - number of various auxiliary file system structures used to arrange files of the mentioned users or groups;
   - other parameters of auxiliary operations performed by the operating system to serve any user or group during a period of time;
   - time and range of modifications of other mentioned consumption parameters allowed for use by consumers to modify limitations already defined;

7. System of calculation of occupied space of Claim 1 that can operate on top of the file systems and does not require modification of a manner data and file metadata are represented as well as the way of representation of the file system service data in the data storage.

Application Area

The given invention is related to the area of methodology of organization of control over the process of space allocation in computer data storages, in particular to organize areas of quoted space.

Level of Technology

Control over the process of organization of data storage is mainly based on major notions and algorithms of support of computer file systems implementation.
File system is part of an operating system intended to provide users with a handy interface when working with data stored on the disk and to provide mutual use of files by several users and processes.
File system conception includes:
- block of all files on the disk,
- sets of data structures used to control files such as file directories, file descriptors, distribution tables for free and occupied space on disk, inode. Please, note that not all file systems require these structures to be stored on disk.

File systems have a rich development history. A traditional simple file system task is to provide access to files. First file systems had only this functionality. Then new tasks, like performance increase, resolved using algorithms of data cashing; also, in that time emerged the tasks of access markers support, fault tolerance, and so on.

One of the main notions of file systems appeared as a result of multi-user mode of computer usage is disk quota [*The Design and Implementation of the 4.4BSD Operating System (Unix and Open Systems Series.) by Marshall Kirk McKusick (Editor), Keith Bostic, Michael J. Karels (Editor)* Addison-Wesley Pub Co; ISBN: 0201549794, p 253-256].

Quotas restrict operating system resources using different criteria: belonging to a user or directory, etc. We are considering quotas that limit disk resources. Disk quotas limit disk space that can be occupied by users of the system. When there is no disk quota distribution system, users can take as much disk space as they want. Such situation may affect system efficiency because other users as well as the operating system itself can get no necessary disk space. The system of disk quota management provides limitation of maximum available disk space that users can get and guarantees that there will be always enough disk space for system operations. Therefore, failure tolerance of the system raises and SLA (Service Level Agreement) is implemented.

In spite of the fact that disk quota management systems are embedded into many operating systems, some OS do not have disk quotas or they are not safely implemented. For example, old versions of Microsoft Windows NT (including version 4.0 and prior) did not have disk a quota management system. There are two main quota types – user quotas and directory quotas. User quotas are based on user criteria; that is, each user has a limit of disk space, file number, etc. Directory quotas are based on directory criteria; that is, each directory has a limit of disk space, file number, etc.

In some known file systems directory quotas are called tree quotas. They limit the number and size of files in the sub-tree of the file system for all users without separating them. This depends on the logical structure of the file systems. In UNIX file systems a file can belong

to different trees, it is necessary to have a new file attribute – tree-id. At that, a file cannot belong to two different trees.

In Microsoft Windows there is a notion of quota objects, which define a control area. Its objects can be a partition, network resource, directory or user. Quota management systems track the activity of data input/output of objects controlling modification of disk space (record, cut, move or delete). If a modification occurs, then data about state of the space occupied by the given object also change. If allowed space is exceeded, then the system undertakes various actions up to interdiction to get disk space by the given object. Additional actions include sending of notification to the user and a record in a log, creation and sending of a report about occupied space, execution of a command file, quota modification.

There are two notions to define quotas: hard quota and soft quota. Hard quota cannot be exceeded. As the user achieves her limits, the resources of the corresponding file system are not allocated any more. For example, if the user has a hard quota of 500 blocks in the file system and currently is using 490 blocks, then the user can get additionally 10 blocks. An attempt to get 11 more blocks will fail.

Soft quotas can be exceeded during some period of time. This period is also called a period of delay or grace period (e.g. a week). If the user exceeds her soft quota during a period of time exceeding the delay, then the soft quota becomes hard and further allocation of resources will be banned. When the user returns to the level below the soft quota, the delay period will be reset.

Mechanism of quota recalculation has various implementations. In some operating systems, a quota to user files is recalculated during authorization of the user, though such approach brings serious time delays. Other implementations provided quota recalculation at some moment of time. But during a time period between recalculation users could seriously exceed allowed quota. The most up-to-date implementation is the one that recalculates a quota during some operations with a file. At that recalculation can be done when an operation is performed by intercepting the operation at the level of the file system driver and analysis of permissibility, and also can be done after the operation is performed. In the letter case the user can considerably exceed her quota by saving a file of big size.

Quota recalculation can be provided using two ways. The first way is when an operating system can review all its files, find the file owned by the user, calculate their size, and make a decision on permissibility of further user file operations. Such way leads to long delays spent to recalculation and practically not implemented for quota analysis per each file operation. Therefore, the second way is more effective, when an operating system (or any other service) keeps records in quota database. Current quotas and the space occupied by each user are stored in the database. For that, it is necessary to track each file operation, that affects the space size and, correspondingly, modify values in the database. Usually, the database is created for one partition and quotas are also set for one partition.

Technical implementation of quotas brings some problems. Among them one of the problems is how to determine file belonging, when it is necessary to determine who among users owns the file, and depending on that check the user quota. Yet file is a logical notion. A physically a file can take less space than logically; that is, it can be compressed by file system tools. Then a question arises: what size should be taken into account when a quota is

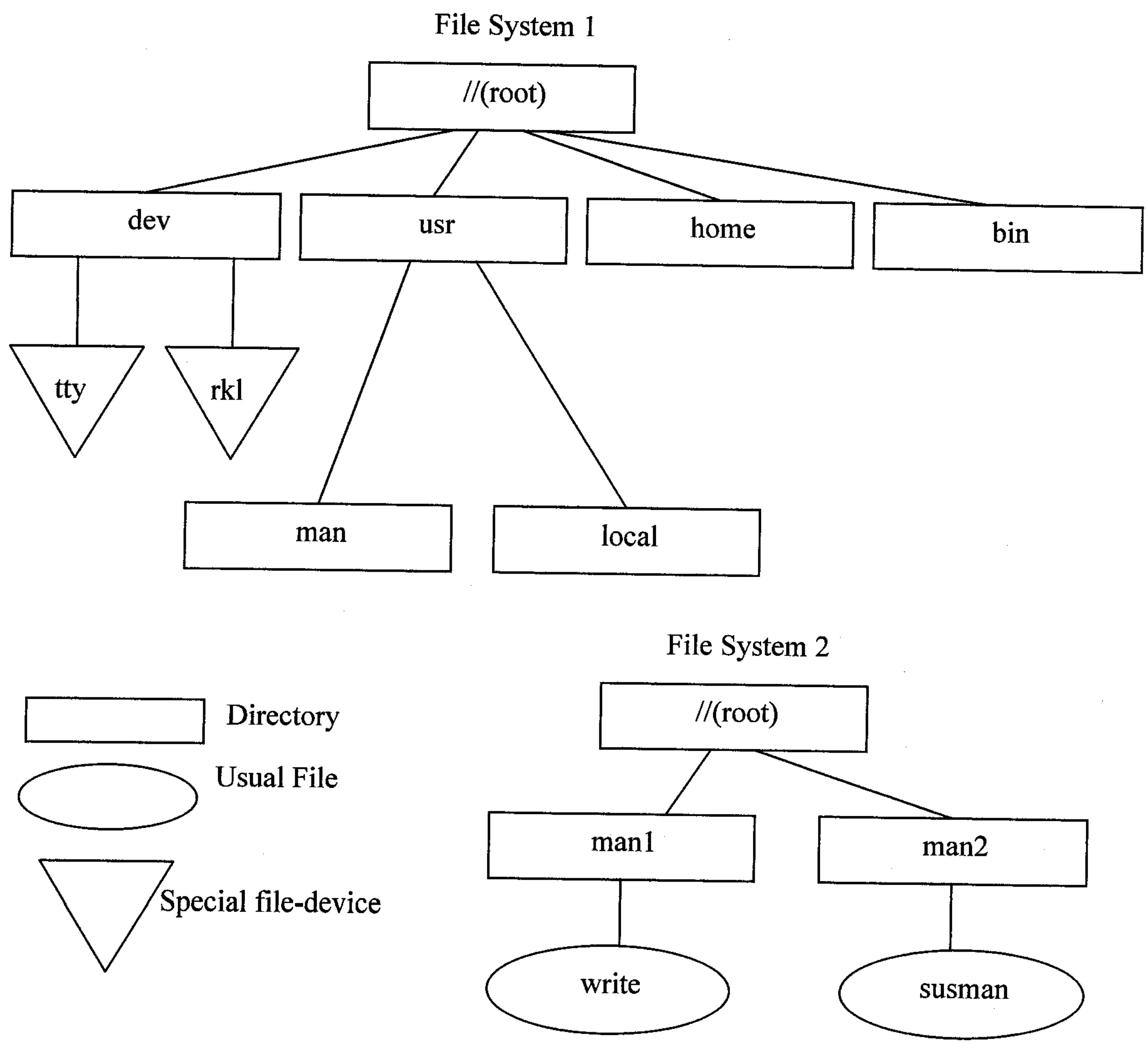
defined as the system has physical disk space constrains. Similarly one file may have several representations – under different names, in different directories, and so on. Physically it takes only one space. Therefore, for correct implementation of quotas it is necessary to take into account various nuances of logical and physical implementation of file systems. A special attention we will attract to the issues of file systems mounting and user accounts.

File System Mounting

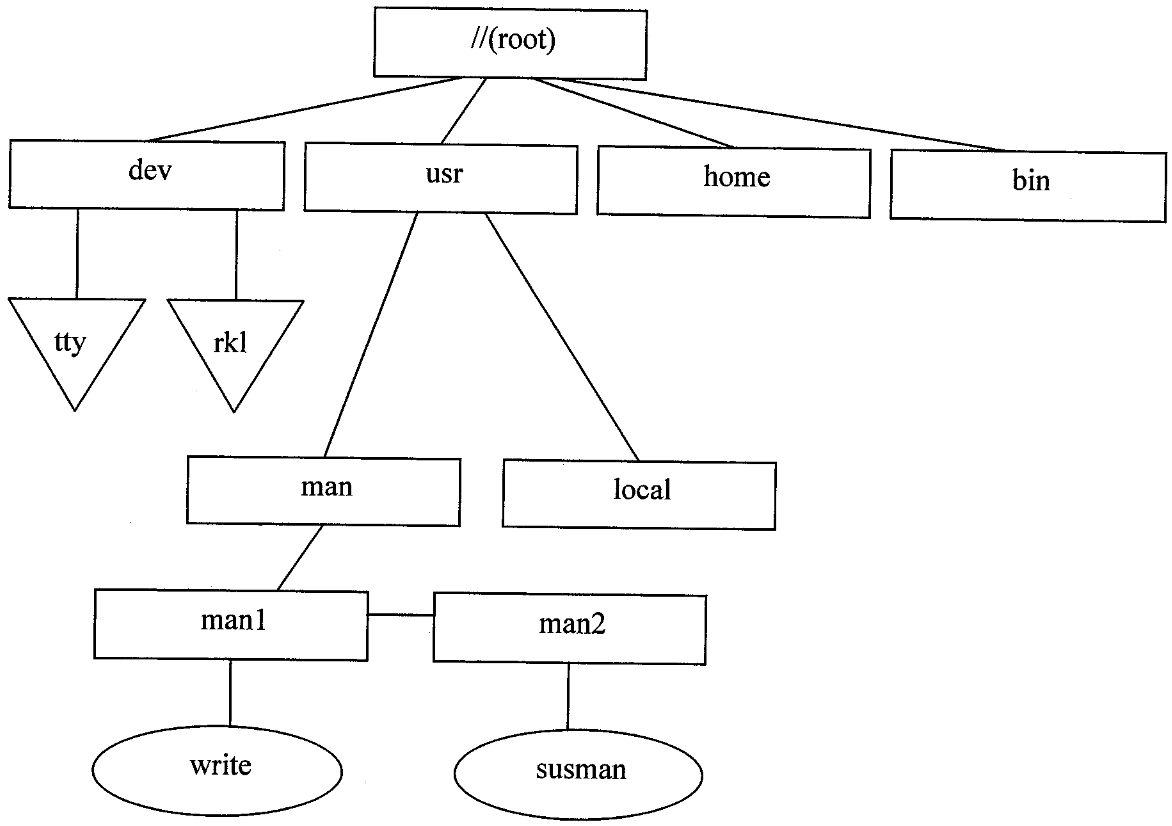
Generally, a computing system may have several disk devices. Even a typical desktop PC has one HDD, one floppy disk, and one CD drive. Powerful computers as a rule have a lot of disk storages with installed disk RAIDs. Moreover, even one physical device may be represented as several logical devices using operating system tools, particularly, by dividing disk space into partitions. But there is question: how to store files in the system, which has several external memory devices.

The first solution consists in that each device contains an autonomous file system; that is, files contained on this device, are described by a directory tree, which is not associated with directory trees on other devices. In this case for unique file identification users must specify the logical device identifier together with the composite symbolic file name. The MS-DOS operating system is an example of such autonomous file systems, where the full file name path includes a letter identifier of the logical disk. Thus, at each access to a file, located on disk A, a user must specify this disk name: A:\privat\letter\uno\let1.doc.

Another variant is to store files so that a user has an opportunity to combine file systems on different devices into the unified file system described by a single directory tree. Such operation is called mounting [Operating Systems: a design-oriented approach/ Charles Crowley. Irwin. 1997. ISBN 0-256-15151-2p 700-703]. Consider how this operation is done using an example with UNIX OS. Among all available logical disk devices the operating system selects one device called a system device. Let's consider two file systems on different logical disks where one disk is a system one.



File system located on the system disk is assigned as a root system. To link file hierarchies in the root system it is necessary to select an existing directory, in our example directory ‘man’. After mounting the selected directory ‘man’ becomes the root directory for the second file system. Via this directory the file system being mounted is attached as a sub-tree to the overall tree.



After mounting of the overall file system for users there is no logical difference between root and mounted file systems; particularly, file naming is provided the same way as if it were unified from the very beginning.

Bind mounting is a mechanism that allows mounting of all or even a part of a file system already mounted to another location. After that the file system becomes available from both mounting points. For example, it is possible to use bind mounting to mount the file system root to directory /home/drobbins/nifty. After that directory /home/drobbins/nifty contains the file system root (/home/drobbins/nifty/etc, /home/drobbins/nifty/opt and so on). If a file in the file system root is modified, then the given modification will be reflected in /home/drobbins/nifty as well. If afterwards the file system is mounted somewhere else (one more bind mounting take place), the all bind mounted and located within the file system being mounted file systems will not be reflected. In other words, if /usr in the root belongs to another file system, then after bind mounting shown above, directory /home/drobbins/nifty/usr will be empty. Therefore, bind mounting allows accessing to the same file via different paths. In some BSD-like OS file systems bind mounting is called **null mounting** [J.-S. Pendry and M. K. McKusick. Union mounts in 4.4BSD-Lite. Conference Proceedings of the USENIX 1995 Technical Conference on UNIX and Advanced Computing Systems (New Orleans), pages 25-33. Usenix Association, 16-20 January 1995].

Loopback mounting is a mechanism that allows mounting of a Linux OS file system stored as an image in a file. For example, command 'mount root_fs.ext2 /mnt.ext2 -o loop,ro'

performs mounting of a file system image 'root_fs.ext2' and provides access to it through directory /mnt.ext2. Such approach is often used to copy CDs when initially a disk image is created in a file and then this image is mapped and works as a usual disk.

User Accounts

Each user and user group usually has a symbolic name as well as a unique numeric identifier. When performed the procedures of user creation, a user provides her symbolic name and password, and an operating system detects the corresponding numeric identifiers of users and groups she is part of. All identification data, including names and identifiers of users and groups, user passwords, as well as information about belonging of users to groups are stored in a special file (e.g., in file /etc/passwd in Unix) or in a special database (in Microsoft Windows NT).

The operating system separates users by assigning them unique in some context identifiers. A domain of computers joined into a network can be considered as such context, e.g., Microsoft Windows Active Directory domain, or a computer system itself, e.g., a standalone computer controlled by a local operating system or any part of the computer system, for example, the 'chroot' environment in an OS like UNIX or virtual environments (VE) provided by Virtuozzo technology. In fact, any number from a certain range can be used as an identifier, e.g., UID of a UNIX-flavored OS user or a more complicated data structure, e.g., Security Identifier (SID) of Microsoft Windows NT.

Practically in all operating systems the access rights matrix for an object can be stored "by parts". For example, in some OSes for each file or directory a so called Access Control List (ACL) is created, where described rights to execute user or user groups operations over this file or directory. File owner ID is stored together with ACL. ACL is part of file or directory characteristics and is stored on a disk in a certain area, for example, in the index descriptor 'inode' of the file system 'ufs'. Not all file systems support ACLs, for example, there are no ACLs in system FAT, because it was developed for the single-user single-program operating system MS-DOS.

In UNIX OS each file has an associated user ID – UID which is unique in OS for each user. In Microsoft Windows NT there is a common object model for each file, which contains such security characteristics as a set of allowed operations, owner identifier, and ACL. An object owner, usually a user who created it, has a right of selective control of access to the object and can modify ACL of the object to permit or forbid others to access the object. An embedded Windows NT administrator unlike a UNIX super-user can not to have some access permissions to the object. To implement this opportunity administrator and administrator group identifiers can be part of ACL as well as identifiers of usual users. However, an administrator can do any operations with any objects, because she can always become an object owner, and then as an owner get the required set of credentials.

An essential part of any modern file system, designed for multi-user access, is a sub-system of disk resources allocation or quota.

To illustrate quota usage let's consider Microsoft Windows NT. Quota management can be implemented using two different approaches. The first approach is used to control file creation and disk space allocation after file operations are over and uses the NT security subsystem to forbid access. The second approach implies integration of a driver into the

operating system, which controls file creation and disk space modification in real time before the input/output operation is over. To forbid access to an application an error "quota exceeded" is displayed.

Patents

1. US patent Document 6,092,163 Kyler, et al. July 18, 2000 711/163
Pageable filter driver for prospective implementation of disk space quotas

Objects of this invention are as follows:
- disk space quotas are implemented in a manner which detects quota violations before information is written to disk.
- a disk I/O operation is failed if it would exceed a quota.
- quotas are applied to open files.
- usage of facilities available in the kernel of the operating system, including synchronization facilities.
- implementation of the invention in pageable code.

The present invention consists in development of a filter driver for implementing disk space quotas. The present invention consists in development of a filter driver for implementing disk space quotas. Quota limits on disk space taken up by files in the file system are established for users and directories, at that an internal database is established to track quotas against actual disk space utilization. A driver uses kernel resources of the operating system to prevent execution of file system I/O operations which would violate any established quota. In doing so, the driver executes a logic in kernel mode which serializes file allocation operations and also serializes access to the internal database.

The first step in this logic is to intercept file system I/O requests before they reach the file system driver. Then the driver determines prospectively (before the I/O request is completed) whether any quota would be exceeded by completion of the I/O request. If a quota would be exceeded, completion of the I/O request is blocked and an error status is issued. If a quota would not be exceeded, the I/O request is allowed to complete and the driver's internal database is updated with revised disk space utilization data.

The invention includes a file system filter driver that has the responsibility of monitoring disk space usage by users, and enforcing the quotas established by the system administrator for each user. Quotas may also be established for directories where files are stored. The invention's file system filter driver intercepts every call bound for the file system driver and processes each of them with respect to their effect on disk space allocation in relation to the established quotas.

The invention keeps a persistent database of the established quotas and the amount of disk space used. This database is updated when disk space changes.

By using a file system filter driver to implement quotas, the invention is able to evaluate the effects of file system operations before the operation is actually executed. This allows the invention to enforce quotas in real time with a high degree of precision. Since the driver

works in the actual I/O path, it can fail I/Os with the appropriate "Quota Exceeded" status code.

2. US patent Document 5,940,841 Schmuck, et al. August 17, 1999 707/205
Parallel file system with extended file attributes

Similar inventions are described in the following patents:
United States Patent 5,946,686 Schmuck, et al. August 31, 1999
Parallel file system and method with quota allocation
United States Patent 5,956,734 Schmuck, et al. September 21, 1999
Parallel file system with a quota check utility

This invention provides a shared disk file system where a file system instance on each machine has identical access to all of the disks coupled to and forming a part in the file system. This can occur using a gateway processor, a switched network, a high speed intranet coupling as would support TCP/IP, a non-uniform memory access bus couplings or other similar connections. In accordance with the given invention, the shared disk file system supports disk read and write calls with associated management calls. The operating system instance is a commonly available or standard and does not need to be changed to use the shared disk file system. The invention has provided new services needed to make the shared disk file system operate in a useful fashion.

The described shared file system operates as a parallel file system in a shared disk environment.

In the UNIX world the Quota concept is well known by that name. It is a generic concept able to be used to manage the initial extent of a space, and this concept is used with other operating systems, such as those of S/390 systems. The invention provides recoverable local shares for the centralized quota management.

Quota Imposition

The main problem with quotas is that they must be directly specified in the whole set of nodes. Though quotas can be controlled from the central server, such solution is not optimal, because in this case the central server would become a bottleneck as each write operation would request permission for execution from the central server. The patent describes the method of issuing of shares of quota to computers that provide active writes to the file system on behalf of the user. Recovery of such resource incase of an error is also considered.

In a parallel file system all disks composing it can be independently interrogated for file read or write. When a processor creates files, it requires a number of sectors allocated on the disk. Sectors allocated to a file would belong to a certain user and be limited by a quota which contains information on the disk space that can be used by a user or group of users. The problem here is that the user can work on several processors simultaneously and be limited by the same quota. Centralization of allocation of new disk blocks replaces a massive parallel file system.

The invention provides a system which allocates shares of quota to each node, re-assigns them according to requirements and recovers them in case of failure. This solution is a method of quota management for disk blocks and inode in the massive parallel computing. The job is shared between one quota server per file system and quota client for a node per file system, which actively processes data within the file system.

Quota limit is a threshold upon which the user can allocate inode or file system space. In the given patent quota is a number of inodes and disk space a user can take. Local share is a space the quota client can allocate for the given user without connecting to the quota server.

The server supports the permanent file which contains quota limits and accumulates data about the usage of file space for all users of the parallel file system. The file is available only on the server, which provides access to the file and update of the file for all processors. That is, only the server has general information about quota usage and space allocation.

All actions related to general quota management are performed on the server. Limit modification, allocation of local shares and current status representation require coordination with the quota server. Quota clients perform modifications in distribution of file space according to the local share and periodically update data on the server depending on the share utilization. The server can annul the client's share to give this share to another client.

Quota clients by default have a zero share. Only if an application on the processor starts creating new data on the file system, the local share would be requested from the server. Only if the client gets the corresponding local share, the application request will be satisfied. Otherwise the request will not be satisfied. The quota client keeps a record of the local share and space taken by it. Applications that free disk space increase local shares for the user. The quota client periodically updates the quota server and provides it with information about local shares usage.

The quota server distributes local shares until it has available quotas; that is, while the system-wide quota limit is not exhausted. If all available quotas are given out for local shares, then the quota server will revoke them to provide new requests. The server will revoke a part of the local share, which allows the user to use the remaining part of the share. The server can revoke the most of local share until no quotas remain, which can lead to denial to the application's request.

Essence of Invention

Our invention provides a method of management of the process of disk space allocation in computer data storages, particularly concerning areas of quotable space.

The suggested method consists in establishment of quotas for an area of the file system space, defined taking into account belonging of the file to a certain subtree or subtrees of the file system.

Quota management is established within a usual computer system which includes the hierarchic file system as well as a set of unique users that could be united into groups.

The software of the computer system separates users by assigning unique in some context identifiers to them. Such context can be implemented through a domain of computers linked into network, for example Microsoft Windows Active Directory domain, or the whole computer system, for example, a standalone computer with the local operating system, or any part of a computer system, for example, the 'chroot' environment in a UNIX OS or Virtual Environments (VE) of Virtuozzo technology. The invention can use as an identifier any number from a certain range, for example, user UID of a UNIX OS, or a more complicated data structure, for example, Security Identifier (SID) of Microsoft Windows XP. Several users with their unique identifiers can make a group, which can also get its own unique among groups identifier; several groups can also make a group with its own identifier, and so on. One user or group can enter into several groups simultaneously; that is, membership in a group is not exceptional.

The file system of the computer system is implemented as a hierarchic tree, where each file has a so called "full access path" and also has user identifiers, and probably one or several groups declaring belonging of this file to this user and corresponding groups. Thus, a computer system program or the operating system kernel can access files of the file system by specifying the access path to the file, and there is always an opportunity to determine belonging of the file to any user and/or groups.

At the computer's boot-up a specific procedure of the initial system rollout is executed which creates an initial root file system that usually contains special files of the operating system intended to serve procedures of the initial system configuring and startup. Later on the expansion of the file system available for operating system processes is provided by mounting of the additional space.

Let us consider the process of file system mounting more thoroughly (Figure 1). The hierarchic file system **100** created already by the beginning of the mounting process starts from some area **101** specified as a root of the file system (though in some systems, for example, in Microsoft Windows, this area is hidden from users' direct access). The purpose of the mounting process **103** is to attach the data storage **102** to the file system directory **104** to let the processes of the computer's operating system access information located at area **102** of the file system the same way as they do to the files located at area **100**. The storage 102 can be a part of the physical disk, for example, a specifically created and formatted partition, or network storage with the corresponding interface providing access, or area inside of the existing file system **100** access to which after such a procedure can be done by an alternative path.

After an area of the file system has been mounted (Figure 2), the processes of the operating system and its kernel acquire an opportunity to get access inside of the file system **204** to the newly mounted subtree **201** of the file system via the mounting point **200**. For example, if area **201** is mounted with the access path /usr/bin, then its internal file **202** will be accessible by the path /usr/bin/X11/xclock.

Let us define the area of computation of the space used by the file system as one or several subtrees of the computer file system where each of subtrees can include only one mounting area. For example, on Figure 3 the file system of computer **300** consists of the base part of system **305** and mounted sub-area **302** with the access path **301** /usr/bin, and the quota area**304** lies on the search path below directory /usr and includes all files of the base area composing **304**, but does not include files lying in the mounting area **302**; that is, for

example, the file /usr/local/bin/gcc belongs into the area of quoting, but files from subdirectory /var or /usr/bin/X11/xclock will not.

Belonging to the quota area is defined by the full access path to the file and this is done during opening of the file. If the full access path contains the path's part belonging to the area of quoting, starting from the directory root, then the file is considered belonging to the area of quoting. Thus, for example, the file /usr/local/bin/gcc on Figure 3 belongs to the quota area304 because its full path contains the part "/usr/" which is describes the area of quoting. Technically, the definition of belonging can be done in a different way – by direct comparison of access paths as well as by other means, for example, by recursive setting of an attribute of belonging of all files of the given subdirectory of the given quota area if the directory itself has such attribute.

For each quota area a set of quota parameters is defined. For example, the standard set of quota parameters in UNIX OS is the size of occupied disk space and number of so called inodes – in fact, the number of files that can be created by a user in the area of quoting. Other values depending on the type of a data storage and file system can be used as quota parameters.

Belonging of the file to a certain user is defined at the moment when it is accessed and depends on the unique user identifiers stored in the file system. The quota management program associates each unique identifier to its own record that stores current values of quota parameters for this area for the corresponding identifier. The record can be stored in different ways – in a separate file, unique for the given area as well as in the database or inside service data of the file system.

The uniqueness of the identifier is to be provided within any context. The entire operating system can be used as such context and this means that all identifiers must be unique inside it. Also, a dedicated area of the operating system, such as 'chroot' or virtual environment can be used as a context. In such situation the same identifiers can be associated with the files that belong to different areas, but the quota management system will consider them separately, taking into accounts only those that are inside of one area. Thus, there is an opportunity to support different users with the same identifiers, which takes off limitations on performance for the selected identifiers inside of one environment.

Quota management system can be both part of the operating system kernel, or a special process. It responsible for receiving of requests on data size change, their analysis depending on the current state of quota parameters as well as for modification of stored parameter values. It must be part of the file system functioning support system controlling the process of space allocation of the data storage.

To allocate new space or change the size of the space already taken, the following sequence of actions is used:
- Definition of belonging of the mentioned space to some file of the file system, for example, if there is a request to increase the size of the existing and open file, it is necessary to determine its name and other parameters pointing to it.
- Definition of belonging to the area of computation of used space as described above.
- Definition of an identifier or identifiers by which quoting is performed. If a file is already created and stored in the storage, then the identifier is defined by a stored with it identifier

of a group and/or user. If a file does not exist, then belonging is defined by identifiers associated with the process, which had initiated creation of this file.
- Search by the defined identifier of quota parameters that can be stored in a special file, database, network server or identifiers.
- Comparison of found limitations with the current value of used data storage space, current state of the computer system, current state of the file system and requested sizes of space allocation and other parameters of the data storage, which can affect the decision to accept or reject an operation. For example, if all the space allowed to a given user is already taken, then an attempt to request a new area of space in the storage will fail.

Let us consider an example of the process of alteration of space size occupied by the file taking into account quotas on the reserved resources (Figure 4). Assume that the file system 400 contains a quota area 402. A process of a user or operating system 401 tries to change the length 405 of the file /usr/local/log/mylog 404 inside of the quota area 402 with the resource accounting file 406. The quota control system 407, which can be implemented as part of the operating system driver, responsible for the file system, or as an autonomous daemon of the operating system, receives this request, defines belonging of the file 404, and gets one or several unique identifiers by which it finds records corresponding to the given identifiers in the quota data file 406. Then the quota control system 407 checks whether this operation is permitted; that is, whether indices of space usage stay within permitted limits after the operation is over, and if the check is positive, allows the procedure of space size alteration 408 to be executed. After the operation has been completed, the file 409 will become modified as well as the record about current values in the quota data file 411. Depending on the method of the quota control system 407 implementation, it is possible that because of bufferization the record about changes appears in the quota data file 411 later the record in the file 409 itself. Quota data file 411 can be not only inside of, but also separately from the data storage and be a database or a special server with network access.

Before usage, the quota data file must be initialized and filled out with initial values of parameters of data storage space usage for unique identifiers and groups inside of the area of used space calculation, defined by a computer system administrator or any other systems of automatic quota control.

Quota parameters depend on the used file system and can be as follows:
- number of block occupied by data of one user partially or entirely for the file system, implemented above the block storage;
- number of structures available to the user for the file system using special data structures associated to a file, for example, number of inodes in UNIX systems;
- number or other parameters of service operations performed by the operating system to serve any user or group of users for a period of time, for example, a number of backup operations, or the size of backup copies, or a number of disk operations done per a period of time;
- time and range of changes of other consumption parameters, allowed for usage by users that modify limits already set up, for example, a user may be allowed to temporarily violate the quota for occupied resources, but for a strictly limited time and within some specific limit.

The proposed quota control system does not require modification of the way files stored in the file storage, including file metadata and the method of representation of service data of the file system; that is, it can work with the file system of any type.

The difference between the invention described in US patent Document 6,092,163 Kyler, et al. July 18, 2000 711/163 "Pageable filter driver for prospective implementation of disk space quotas" and the way it manages quotas using filter drivers and our invention is that it works on the level of disk data storage blocks, but not on the level of files and do not use an archive bit, used as an attribute when archiving is required.

The difference between the invention described in US patent Document 5,940,841 Schmuck, et al. August 17, 1999 707/205 "Parallel file system with extended file attributes" and our invention is that it quotes data on the level of the tree of one file system without passing to other volumes (that is, inside of one mounting point), defining the necessity for quoting by the file access path, but not by the data storage volume, within the local computer file system.

The difference between the software like "QuotaAdvisor" by Precise Software Solutions and the way it manages quotas for directorie and our invention is that it does not establishes a quota for the whole subtree size, but establishes separate quotas for different users, who have their files in the shown directory.

Figures

Computer file system before mounting of the subtree;
directory "/usr/bin" is empty

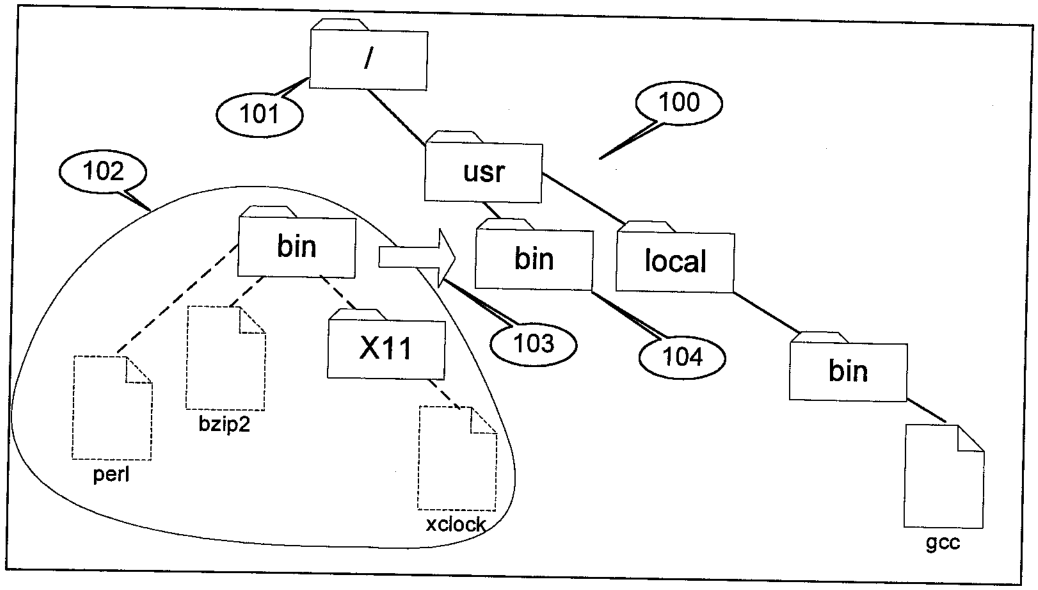


Figure 1.

Computer file system after mounting of the subtree to directory "/usr/bin"

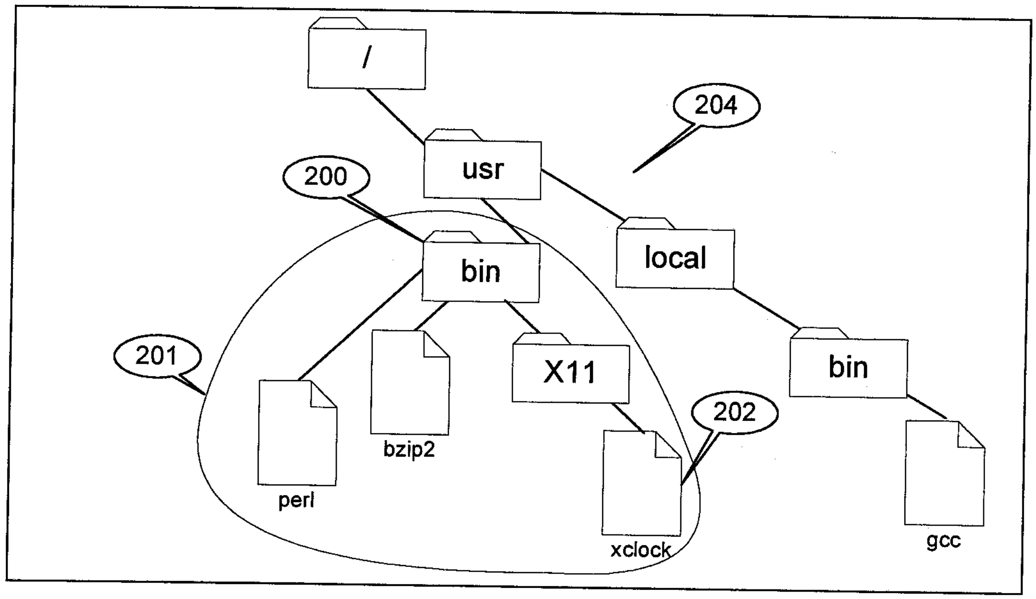


Figure 2.

Area of calculation of space used by the file system

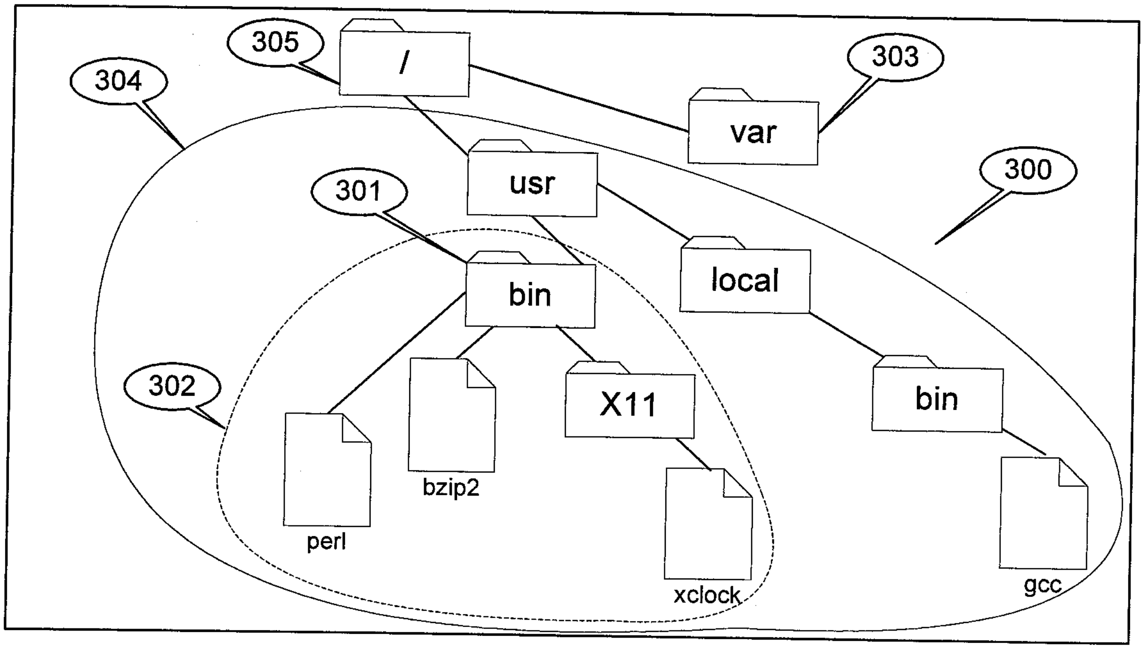


Figure 3.

Process of alteration of space size occupied by the file taking into account a quota on the reserved resources

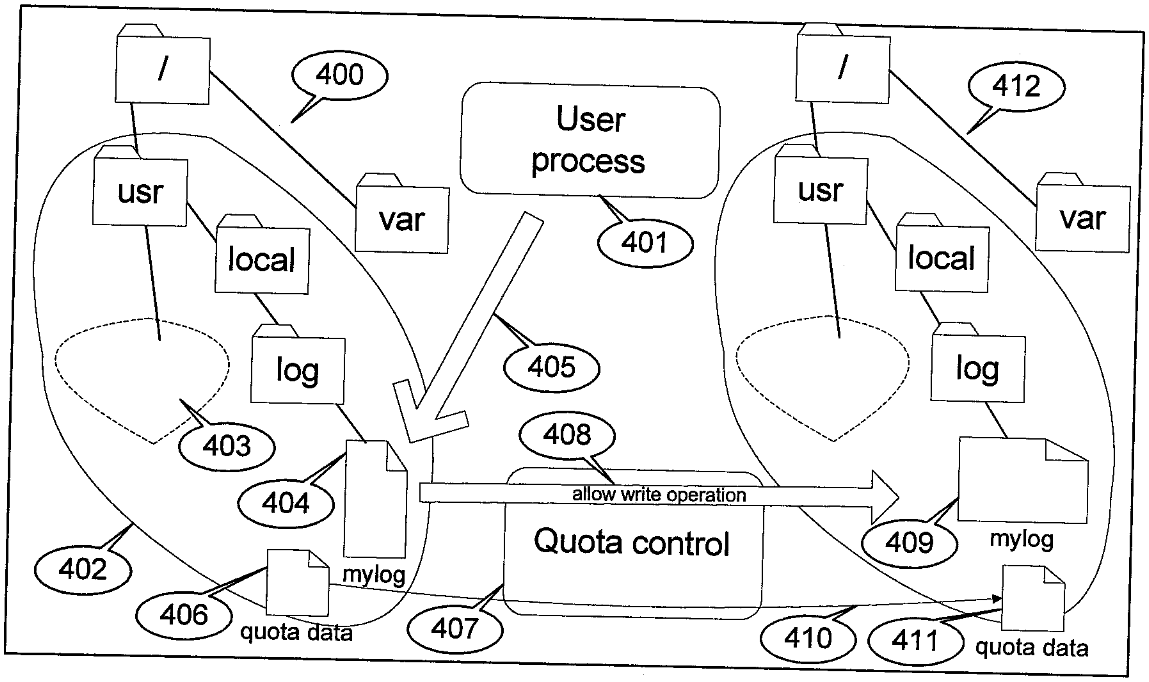


Figure 4.